PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-015647
(43)Date of publication of application : 22.01.1999

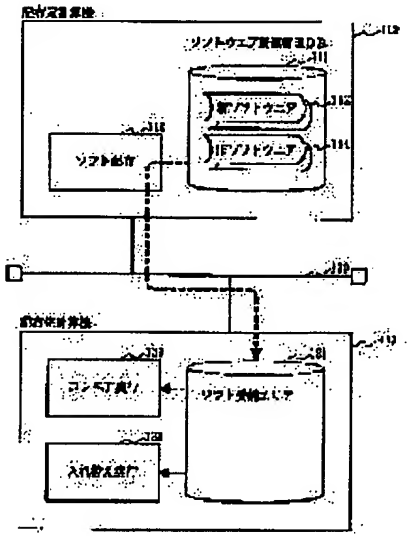
(51)Int.Cl. G06F 9/06
G06F 9/06

(21)Application number : 09-169939
(22)Date of filing : 26.06.1997

(71)Applicant : HITACHI LTD
(72)Inventor : NISHIJIMA HIDEJI
FUJIWARA KAZUNORI
KONO KATSUMI
WATAYA HIROSHI

(54) SOFTWARE MAINTENANCE METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a means for ensuring the restoration of the software of a former (present) version for the failure of the exchange of software from the present version to a new version in the maintenance of software.
SOLUTION: A comparison execution means 132 is provided for a distribution destination computer in a state where the new version 113 and the former version 114 of software resource are stored in the prescribed storage area 131 of the distribution destination computer. The comparison execution means verifies identity between an execution module which is artificially generated from the software of the former version and an execution module (that is, the feasible form of the software of former version) which is previously registered before the exchange of software from the former version to the new version. When a verified result is satisfactory, a processing 133 for exchanging software to that of new version is executed. When the verified result is not satisfactory, the exchange processing 133 is stopped.



LEGAL STATUS

[Date of request for examination] 26.05.2004

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平11－15647

(43)公開日　平成11年(1999)１月22日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/06	４１０	Ｇ０６Ｆ　9/06	４１０Ｐ
	５４０		５４０Ｃ

審査請求　未請求　請求項の数12　ＯＬ　（全９頁）

(21)出願番号　特願平9－169939

(22)出願日　平成９年(1997)６月26日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　西島　英児
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(72)発明者　藤原　和紀
茨城県日立市大みか町五丁目２番１号　株式会社日立製作所大みか工場内

(72)発明者　河野　克己
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

(74)代理人　弁理士　小川　勝男

最終頁に続く

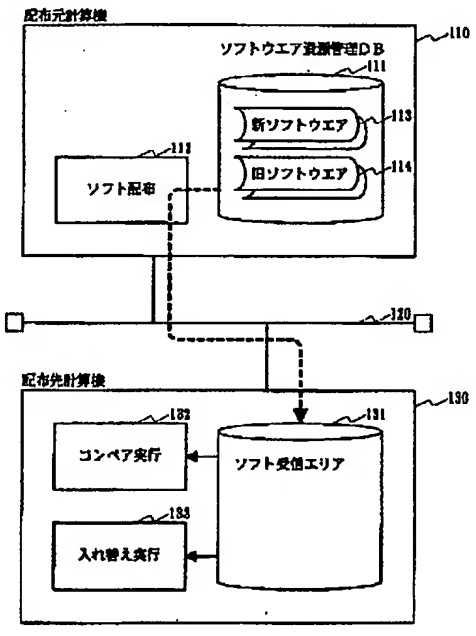
(54)【発明の名称】　ソフトウエア保守方法

(57)【要約】

【課題】ソフトウエア保守において、現行バージョンから新バージョンへのソストウエアの入れ替えの失敗に備えて、旧（現行）バージョンのソフトウエアへの復旧を保証（常に元通りに戻せられるということ）しておく手段を提供することである。

【解決手段】ソフトウエア資源の新バージョン１１３と旧バージョン１１４が配布先計算機の所定の記憶領域１３１に格納された状態において、配布先計算機にコンペア実行手段１３２を設けておき、該ソフトウエアを旧バージョンから新バージョンに入れ替える前に、コンペア実行手段が旧バージョンのソフトウエアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウエアの実行可能形式を指す）間の同一性を検証する。検証結果が良好の時には、新バージョンのソフトウエアに入れ替える処理１３３に移行するが、検証結果が不良の時には、このような入れ替え処理１３３を中止する。

図1



【特許請求の範囲】
【請求項１】配布元計算機が新バージョンのソフトウエア資源を配布先計算機へ格納し、該配布先計算機が新バージョンと旧バージョンのソフトウエア資源を保持しており、該ソフトウエア資源の各バージョンがソースファイルまたはオブジェクトまたはロードモジュールおよびインストールスクリプトまたはコンペアスクリプトからなり、該ソフトウエア資源のうちのインストールスクリプトを起動して実行可能形式を生成するための入れ替え実行手段を有する場合において、該配布先計算機にコンペア実行手段を設けて、該入れ替え実行手段が起動する前に、該コンペア実行手段が該コンペアスクリプトを起動することによって該旧バージョンのソフトウエア資源から生成される実行可能形式と現行稼働状態の実行可能形式間の一致性を検証した後に、検証結果に従って該入れ替え実行手段が実行可能形式の入れ替えを実施することを特徴とするソフトウエア保守方法。
【請求項２】請求項１項記載の配布先計算機において、前記コンペア実行手段が該インストールスクリプトを解析してコンペアコマンドを生成・実行することによって該旧バージョンのソフトウエア資源から生成される実行可能形式と現行稼働状態の実行可能形式間の一致性を検証した後に、検証結果に従って該入れ替え実行手段が実行可能形式の入れ替えを実施することを特徴とするソフトウエア保守方法。
【請求項３】請求項１項記載の配布先計算機において、前記コンペア実行手段がインストール先を一時的なエリアに指定して該インストールスクリプトを起動した後に、該一時的なエリアに生成された実行可能形式と現行稼働状態の実行可能形式間の一致性を検証した後に、検証結果に従って該入れ替え実行手段が実行可能形式の入れ替えを実施することを特徴とするソフトウエア保守方法。
【請求項４】前記コンペア実行手段が配布元計算機から配布先計算機へソフトウエア資源を格納した後に起動されること、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【請求項５】前記コンペア実行手段が配布先計算機の立ち上げ後に起動されること、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【請求項６】前記コンペア実行手段が配布先計算機の立ち下げ時に起動されること、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【請求項７】前記コンペア実行手段が指定された日付や時刻で起動されること、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【請求項８】前記コンペア実行手段が低い優先度で起動されること、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【請求項９】前記コンペア実行手段による検証結果が全て一致の時に限り前記入れ替え実行手段を実行し、該検証結果が少なくとも１つの不一致の時には前記入れ替え実行手段を実行しないこと、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【請求項１０】前記コンペア実行手段による検証結果が一致となったソフトウエア資源に対して前記入れ替え実行手段を実行すること、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【請求項１１】前記コンペアスクリプトが前記フトウエア資源を元に一時的に生成する実行可能形式と現行稼働状態の実行可能形式間の一致性を比較するコマンド郡からなること、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【請求項１２】前記インストールスクリプトが前記ソフトウエア資源を元に実行可能な形式を生成するコマンド郡からなり、ソフトウエア種別・ソフトウエア名・ソフトウエア資源名・インストールオプション別に分類されたフォーマットからなること、を特徴とする請求項１～３項記載のソフトウエア保守方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、分散配置された複数の配布先計算機へのソフトウエア保守に係り、特に、各配布先計算機へ新バージョンのプログラムを入れ替える前に、万が一入れ替えが失敗した時にシステム稼働の異常事態を避けるために、元の状態に確実に復旧できるという保証をとっておくのに好適なソフトウエア保守方法に関する。
【０００２】
【従来の技術】従来の技術は、特開平第５－１０８３１７号「ネットワークシステム及びそのソフトウエア管理方法」に示されている。
【０００３】従来の技術は、配布元計算機が事前に準備された新バージョンのソフトウエアを保持しており配布先計算機に新バージョンの格納エリアにダウンロードした後に、配布先計算機が新バージョンの格納エリア内のソフトウエアのうちのインストールスクリプトを起動することによって現行バージョンから新バージョンへのソストウエアの入れ替えを実施する。新バージョンへのソストウエアの入れ替えが成功した時には、新バージョンの格納エリアを現行バージョンの格納エリアに移動させる。一方、新バージョンへのソストウエアの入れ替えが失敗した時には、配布先計算機が旧（現行）バージョンの格納エリア内のソフトウエアのうちのインストールスクリプトを起動することによって旧（現行）バージョンのソフトウエアへの復旧を実施する。常に、旧（現行）バージョンの格納エリアには、前回の入れ替えの実施で成功した、旧（現行）バージョンのソフトウエアが格納されている。
【０００４】

【発明が解決しようとする課題】従来の技術は、万が一現行バージョンから新バージョンへのソストウエアの入れ替えが失敗した時に、旧（現行）バージョンのソフトウエアへの復旧が保証（常に元通りに戻せられるということ）されていないために、旧（現行）バージョンのソフトウエアへの復旧も失敗することがあり、結局、配布先計算機のソフトウエアシステムの動作不能によって業務の支障を招く恐れがある。具体的な一例を以下に説明する。

【０００５】例えば、リアルタイム制御計算機に代表される常駐型タスクは、主メモリ内にローディングされることによってインストールが完了するものであり、次のようなインストールの制約がある。なお、タスクとは同時に起動できる実行モジュール数が１つに限られている実行モジュールの種別を指す。

【０００６】（Ａ１）メモリ内の実行モジュールをバックアップする手段がない

（Ａ２）メモリ内の他の実行モジュール間とリンケージをとるためのインストールスクリプトが必要となる。

【０００７】このようなインストールの制約によって、次のような問題点がある。

【０００８】「タスクを新バージョンに入れ替えた時に、これが万が一失敗した時に、元の旧バージョンへ確実に戻すことができない」この問題の発生する原因には、旧バージョンの入れ替え直後と新バージョンの入れ替え直後間で状況の変化があるためで、具体的には次のような要因があげられる。

【０００９】（Ｂ１）ローディングパラメータの違い

（Ｂ２）メモリ使用状況の変化

（Ｂ３）パッチによるメモリ空間のアドレスとデータ関係の変化

このような要因がある限り、ソフトウエア保守の高信頼化を図るためには、次の課題を解決する必要がある。

【００１０】「メモリ内へ常駐する実行モジュールが、入れ替える前の状態と、復旧した後で、アドレス空間およびデータが一致化されていることを保証しなければならない」

以上のように、本発明の目的は、現行バージョンから新バージョンへのソストウエアの入れ替えの失敗に備えて、旧（現行）バージョンのソフトウエアへの復旧を保証（常に元通りに戻せられるということ）しておく手段を提供することである。

【００１１】

【課題を解決するための手段】ソフトウエアの新バージョンと旧バージョンが事前に配布先計算機の所定の記憶領域に格納された状態において、配布先計算機にコンペア実行手段を設けておき、該ソフトウエアを旧バージョンから新バージョンに入れ替える前に、コンペア実行手段が旧バージョンのソフトウエアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウエアの実行可能形式を指す）間の同一性を検証する。

【００１２】検証結果が良好の時には、新バージョンのソフトウエアに入れ替える処理に移行するが、検証結果が不良の時には、このような入れ替え処理を中止する。

【００１３】コンペア実行手段は、旧バージョンのソフトウエアがソースファイルまたはオブジェクトやライブラリおよびインストールスクリプトに加えてコンペアスクリプトを含んでいる時には、コンペアスクリプトを起動する。または、旧バージョンのソフトウエアがソースファイルまたはオブジェクトやライブラリおよびインストールスクリプトのみを含んでいる時には、インストールスクリプトを解析して、ソフトウエア種別に従ってコンペアコマンドを生成・実行する。ここで、コンペアスクリプトとは、旧バージョンのソフトウエアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール間を比較するためのコンペアコマンドを記述したものである。インストールスクリプトとは、旧バージョンのソフトウエアから実行モジュールを生成するためのコンパイル・リンク・ロードコマンドを記述したものである。

【００１４】ソフトウエアが複数ある場合には、コンペア実行手段は、複数のソフトウエアを実行対象とする。

【００１５】前記コンペア実行手段によって、旧バージョンのソフトウエアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウエアの実行可能形式を指す）間の同一性を検証しておくため、万が一、該ソフトウエアを旧バージョンから新バージョンに入れ替える処理が失敗した時でも、旧バージョンのソフトウエアへの復旧処理は、入れ替える前の実行モジュールに元通りに戻すことが可能となる。

【００１６】このような元通りに戻せられるという高信頼なソフトウエア保守の実現によって、常にソフトウエアは正常な動作状態を維持することができ、ソフトウエアシステムの安全な稼働を保証させることができる。

【００１７】

【発明の実施の形態】以下、本発明を実現するための実施例を図面により説明する。

【００１８】［１］実施例１

（Ａ）ハード・ソフトウエア構成例

図１に本発明を実現するためのハード・ソフトウエア構成の一例を示す。図１は、ハードウエア構成として、配布元計算機１１０と配布先計算機１３０がネットワーク１２０に接続された構成を示している。配布先計算機１３０は１つに限定されるものではなく複数のものがネットワーク１２０に接続されていてもかまわない。配布元計算機１１０はソフトウエア資源管理データベース１１１およびソフト配布手段１１２を有する。一方、配布先計算機１３０はソフト受信エリア１３１、コンペア実行

手段１３２、および、入れ替え実行手段１３３を有する。ソフトウエア保守を実施する場合、ソフト配布手段１１２がソフトウエア資源管理データベース１１１のうち入れ替えるべきソフトウエアの新バージョン１１３と旧バージョン１１４を保守対象となる配布先計算機１３０のソフト受信エリア１３１にダウンロードする。旧バージョンのソフトウエアとは、配布先計算機にとって現行バージョンのソフトウエアを指す。新バージョンおよび旧バージョンのソフトウエアとして、それぞれがソースファイルまたはオブジェクトやライブラリおよびインストールスクリプトに加えてコンペアスクリプトを有している。そして、入れ替え実行手段１３３がソフト受信エリア１３１にダウンロードされた新バージョンのソストウエアのインストールスクリプトを起動することによって旧（現行）バージョンから新バージョンへソフトウエアの入れ替えを実施する。もしも、この入れ替えが失敗した時には、入れ替え実行手段１３３がソフト受信エリア１３１にダウンロードされた旧バージョンのソストウエアのインストールスクリプトを起動することによって旧（現行）バージョンのソフトウエアへの復旧を実施する。ここで、インストールスクリプトとは、それぞれのバージョンのソフトウエアから実行モジュールを生成するためのコンパイル・リンク・ロードコマンドを記述したものである。

【００１９】本発明の特徴は、入れ替え実行手段１３３が動作する前に、コンペア実行手段１３２が旧バージョンのソフトウエアのコンペアスクリプトを起動することによって、旧バージョンのソフトウエアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウエアの実行可能形式を指す）間の同一性を検証するものである。ここで、コンペアスクリプトとは、旧バージョンのソフトウエアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール間を比較するためのコンペアコマンドを記述したものである。

【００２０】（Ｂ）配布先計算機のＤＢ構成例と処理の流れ

図２は、ソフト受信エリア１３１のデータベース構成例である。ソフト受信エリア１３１は、ソフト配布手段１１２によってダウンロードされたソフトウエアの新バージョン１１３と旧バージョン１１４が格納される記憶領域である。新バージョンのソフトウエアが「ソフトウエア名.N」のディレクトリ名称の構成下に、一方、旧バージョンのソフトウエアが「ソフトウエア名.B」のディレクトリ名称の構成下に格納される。例えば、入れ替えるソフトウエアとしてtaskAとtaskBが対象の場合、taskA.Nのディレクトリ下２１０にソフトウエアtaskAの新バージョン、taskA.Bのディレクトリ下２２０にソフトウエアtaskAの旧バージョン、taskB.Nのディレクトリ下２３０にソフトウエアtaskBの新バージョン、および、taskB.Bのディレクトリ下２４０にソフトウエアtaskBの旧バージョンが格納される。新バージョンと旧バージョンのソフトウエアは、それぞれがソースファイルまたはオブジェクトやライブラリおよびインストールスクリプトに加えてコンペアスクリプトを有している。例えば、ソフトウエアtaskAの新バージョンはオブジェクトtaskA.o２１１、インストールスクリプトap_load２１２、および、コンペアスクリプトap_comp２１３、ならびに、ソフトウエアtaskAの旧バージョンはオブジェクトtaskA.o２２１、インストールスクリプトap_load２２２、および、コンペアスクリプトap_comp２２３からなる。ここで、オブジェクトtaskA.o２１１とtaskB.o２２１間は新旧間で修正が加えられているので部分的に異なる。また、インストールスクリプトap_load２１２とap_load２２２間およびコンペアスクリプトap_comp２１３とap_comp２２３間はソフトウエア名称やコマンド名称等の変更が稀であるので同一のケースが多い。

【００２１】図３は、コンペアスクリプトap_comp２２３のコンペアコマンドの記述例である。コマンド例としてＵＮＩＸシステムにおけるBourne shを用いている。コンペアスクリプトap_comp２２３が起動されると、３０１、３０２、３０３の順に解析され、結局、解析結果の「comp +P -o taskA taskA.o」が実行される。この「comp +P -o taskA taskA.o」は、オブジェクトtaskA.oから疑似的に生成した実行モジュールtaskAと既に登録されてある実行モジュールtaskA（現行バージョンのソフトウエアの実行可能形式taskAを指す）間の同一性を比較して、両者間の異なる部分を表示する。

【００２２】図４は、コンペア実行手段１３２の処理の流れを示す。ステップ４０１は、ソフト受信エリアから現行バージョンのソフトウエア名の取り出しを行なう。例えば、ソフト受信エリア１３１のうちディレクリ名がサフィックス'.B'を有するものを検索すれば、taskA.B２２０とtaskB.B２４０を取り出すことができる。ソフトウエア名が複数存在する場合には、プログラム名称の順に、以下のステップを実行していく。ここで以下では、ソフトウエア名としてtaskAを用いて説明する。ステップ４０２は、取り出したソフトウエア名を利用者に知らせるために画面等に表示する。例えば、taskA.Bからサフィックス'.B'を取り除いたものをソフトウエア名称として、画面に「ソフトウエア［taskA］のコンペア実行」と表示する。図５が画面に表示される内容の例である。ここでは、図５の５０１のように表示される。ステップ４０３は、取り出したソフトウエア名に対するコンペアスクリプトap_compを起動する。例えば、ソフトウエアtaskAのコンペアスクリプトap_comp２２３を起動する。コンペアスクリプトap_comp２２３を起動すると、結局、「comp +P -o taskA taskA.o」が実行される。この「comp +P -o taskA taskA.o」の実行結果は、オブジェクトtaskA.o２２１から疑似的に生成した実行

モジュールtaskAと既に登録されてある実行モジュールtaskA（現行バージョンのソフトウエアの実行可能形式taskAを指す）間が同一の場合には画面に何も表示されないが、逆に異なる場合には違いの詳細情報が画面に表示される。例えば、もしもこの実行結果が異なる場合には図５の５０２のように表示される。ステップ４０４は、取り出したソフトウエアのうちまだ未処理のものが存在するかを確認して、存在する場合にはステップ４０２に戻り、存在しない場合には終了する。例えば、取り出したソフトウエアのうちtaskBが未処理であるので、ソフトウエアtaskBに対してステップ４０２～４０３を処理する。取り出したソフトウエアに対して全て処理した後に、本処理は終了する。

【００２３】以上のようなコンペア実行手段１３２によって、入れ替えるべきソフトウエアを旧バージョンからから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウエアの実行可能形式を指す）間の同一性を検証した後に、検証結果が良好の時には、新バージョンのソフトウエアに入れ替える処理に移行するが、検証結果が不良の時には、このような入れ替え処理を中止する。このように、同一性を検証しておくため、万が一、入れ替えるべきソフトウエアを旧バージョンから新バージョンに入れ替える処理が失敗した時でも、旧バージョンのソフトウエアへの復旧処理は、入れ替える前の実行モジュールに元通りに戻すことが可能となる。

【００２４】［２］実施例２

実施例１の変形例として、コンペア実行手段１３２がコンペアスクリプトを実行する代わりに、コンペア実行手段１３２がインストールスクリプトを解析して、ソフトウエア種別に従ってコンペアコマンドを生成・実行する実施例を説明する。このようなコンペア実行手段は、インストールスクリプトを元にコンペアコマンドを生成・実行するので、コンペアスクリプトを作成するのがユーザにとって面倒な場合に有効となり、コンペアスクリプトを準備する必要がなくなる。なお、ハードウエア・ソフトウエア構成は図１と同様であり、ソフト受信エリアのデータベース１３１は、図２においてコンペアスクリプトap_comp２１３と２２３がない場合の構成となる。

【００２５】図６は、インストールスクリプトap_load２２２のインストールコマンドの記述例である。コマンド例としてＵＮＩＸシステムにおけるBourne shを用いている。インストールスクリプトap_load２２２が起動されると、６０１、６０２、６０３、６０４の順に解析され、結局、解析結果の「reload +P taskA 100 +L taskA.o -I 10」が実行される。この「reload +P taskA 100 +L taskA.o -I 10」は、オブジェクトtaskA.oのリンケージを実施することによって実行モジュールtaskAがタスク番号100の優先度10で主メモリ内に生成される。

【００２６】図７は、コンペア実行手段１３２がインストールスクリプトを解析して、ソフトウエア種別に従ってコンペアコマンドを生成・実行する処理の流れを示す。図７におけるステップ４０１、４０２、および、４０４は、図４と同様の処理である。ステップ７０１は、取り出したソフトウエア名に対するインストールスクリプトap_loadを解析する。例えば、ソフトウエアtaskAのインストールスクリプトap_load２２２を解析する。ソフトウエア種別によって生成すべきコンペアコマンドが異なるので、ここでは、インストールスクリプトap_load２２２のうちでソフトウエア種別を識別するための変数KINDを設けている。インストールスクリプトap_load２２２を解析すると、変数KINDの値がtaskであるので「comp +P」を生成して、変数NAMEの値がtaskAであるので「-o taskA」を生成して、変数OBJECTの値がtaskA.oであるので「taskA.o」を生成する。これらを並べた結果として、コンペアコマンド「comp +P -o taskA taskA.o」を生成する。ステップ７０２は、生成したコンペアコマンドを起動する。例えば、コンペアコマンド「comp +P-o taskA taskA.o」を起動する。ステップ７０１と７０２は、結果的にはステップ４０３と同様の内容となる。

【００２７】以上のようなコンペア実行手段によって、インストールスクリプトを元にコンペアコマンドを生成・実行するので、コンペアスクリプトを準備する必要がなくなり、ユーザの負担軽減およびディスク容量の削減につながる。その反面、コンペアコマンドを解析して生成するのに要する時間が増加する。

【００２８】［３］実施例３

実施例１の変形例として、コンペア実行手段１３２がコンペアスクリプトを実行する代わりに、コンペア実行手段１３２がインストール先を変更した形式でインストールスクリプトを実行後に、変更したインストール先に生成された実行モジュールと現行バージョンの実行モジュール間でコンペアを実行する実施例を説明する。このようなコンペア実行手段は、ソフトウエア種別がプロセスやテキストデータの場合に有効であり、疑似的に生成した実行モジュールと現行バージョン間の実行モジュール間を比較するコマンドが提供されていない場合に必要となる。なお、ハードウエア・ソフトウエア構成は図１と同様であり、ソフト受信エリアのデータベース１３１は、図８に示す通りである。

【００２９】図８のソフト受信エリア１３１は、入れ替えるソフトウエアとしてprocessAとprocessBが対象の場合を例にしているので、図２のデータベース構成のうちで、taskAがprocessAに、および、taskBがprocessBに置き換わった構成となる。しかも、コンペアスクリプトap_compを含んでいない。すなわち、processA.Nのディレクトリ下８１０にソフトウエアprocessAの新バージョン、processA.Bのディレクトリ下８２０にソフトウエアprocessAの旧バージョン、processB.Nのディレクトリ下

８３０にソフトウエアprocessBの新バージョン、および、processB.Bのディレクトリ下８４０にソフトウエアprocessBの旧バージョンが格納された構成である。ソフトウエアprocessAの新バージョンはオブジェクトprocessA.o８１１およびインストールスクリプトap_load８１２、ならびに、ソフトウエアprocessAの旧バージョンはオブジェクトprocessA.o８２１およびインストールスクリプトap_load８２２からなる。

【００３０】図９は、インストールスクリプトap_load８２２のインストールコマンドの記述例である。インストールスクリプトap_load８２２が起動されると、９０１、９０２、９０３、９０４の順に解析され、結局、解析結果の「cc -o /usr/local/bin/processA processA.o」が実行される。この「cc -o /usr/local/bin/processA processA.o」は、オブジェクトprocessA.oのリンケージを実施することによって実行モジュールprocessAがインストール先の/usr/local/binに生成される。

【００３１】図１０は、コンペア実行手段１３２がインストール先を変更した形式でインストールスクリプトを実行後に、変更したインストール先に生成された実行モジュールと現行バージョンの実行モジュール間でコンペアを実行する処理の流れを示す。図１０におけるステップ４０１、４０２、および、４０４は、図４と同様の処理である。ステップ１００１は、インストール先に正式に実行モジュールを生成せずに仮の実行モジュールを必要とするために、取り出したソフトウエア名に対するインストールスクリプトap_loadをインストール先を変更して実行する。例えば、ソフトウエアprocessAのインストールスクリプトap_load８２２のうちで指定されたインストール先の変数INSTDIRの値を"."（カレントディレクトリを意味する）に変更した結果、すなわち、「cc -o ./processA processA.o」を実行する。この結果、カレントディレクトリ（図８の８２０）に実行モジュールprocessAが生成される。ステップ１００２は、仮に生成した実行モジュールと正式な現行バージョンの実行モジュール間で比較検証を行なう。例えば、ＵＮＩＸシステムにおけるファイル比較コマンドcmpを利用して、「cmp ./processA /usr/local/bin/processA」を実行することによって比較検証を行なう。正式な現行バージョンの実行モジュールの所在は、インストールスクリプトap_load８２２の変数INSTDIRの値（/usr/local/bin）である。このように、ステップ１００１と１００２は、インストールスクリプトap_load８２２を元に、仮の場所に生成させた実行モジュールと正式な現行バージョンの実行モジュール間の比較検証を行なうことができる。

【００３２】［４］各実施例の補足

（Ａ）コンペア実行手段の起動タイミング

実施例１～３のそれぞれにおいて、コンペア実行手段１３２の起動タイミングとして、（１）配布先計算機に入れ替えるべきソフトウエアをダウンロード後、（２）配布先計算機を立ち上げ後、（３）配布先計算機を立ち下げ時、（４）ユーザの指定した日付と時刻、（５）ＣＰＵ負荷が低い時、等を設けておき、配布先計算機の特徴に応じて使い分ける。上記の（１）は、ソフト配布手段１１２が入れ替えるべきソフトウエアをダウンロードした後にコンペア実行手段１３２に実行指令を与えれば実現される。上記の（２）は、配布先計算機の立ち上げのイニシャル処理の中にコンペア実行手段１３２に実行指令を要求するコマンドを記述することによって実現される。上記の（３）は、配布先計算機の立ち下げのシャットダウン処理の中にコンペア実行手段１３２に実行指令を要求するコマンドを記述することによって実現される。上記の（４）は、ソフト配布手段１１２が入れ替えるべきソフトウエアをダウンロードした後にコンペア実行手段１３２に指定日付・時刻での実行指令を与えれば実現される。指定日付・時刻での実行指令にはＵＮＩＸシステムでのatコマンドがある。上記の（５）は、ソフト配布手段１１２が入れ替えるべきソフトウエアをダウンロードした後にコンペア実行手段１３２に低優先度指定での実行指令を与えれば実現される。低優先度指定での実行指令にはＵＮＩＸシステムでのniceコマンドがある。このような起動タイミングの使い分けによって、適切なタイミングでソフトウエア保守を実施できるので、円滑な計算機システムの運用を達成できる。

【００３３】（Ｂ）コンペア実行から入れ替え実行への移行

実施例１～３のそれぞれにおいて、コンペア実行手段１３２の処理終了後から入れ替え実行手段１３３への移行の仕方として、（１）比較検証結果が全て一致の時に限り入れ替え実行へ移り、少なくとも１つの不一致がある時には入れ替え実行を中断する、（２）比較検証結果が一致のソフトウエアのみを入れ替え実行手段が実施し、不一致のソフトウエアを無視する、等の手段を設けて、配布先計算機の特徴に応じて使い分ける。このような移行方法の使い分けによって、計算機システムの信頼度に従ったソフトウエア保守を実施できる。

【００３４】

【発明の効果】以上のような、コンペア実行手段によって、旧バージョンのソフトウエアから疑似的に生成される実行モジュールと既に登録されてある実行モジュール（旧バージョンのソフトウエアの実行可能形式を指す）間の同一性を検証しておくため、万が一、該ソフトウエアを旧バージョンから新バージョンに入れ替える処理が失敗した時でも、旧バージョンのソフトウエアへの復旧処理は、入れ替える前の実行モジュールに元通りに戻すことが可能となる。

【００３５】このような元通りに戻せられるという高信頼なソフトウエア保守の実現によって、常にソフトウエアは正常な動作状態を維持することができ、ソフトウエアシステムの安全な稼働を保証させることができる。

【図面の簡単な説明】

【図１】ソフトウエア保守システム構成の一例である。

【図２】ソフト受信エリアのデータベース構成（タスクの対象）の一例である。

【図３】タスク向けコンペアスクリプトのコマンド記述の一例である。

【図４】コンペア実行手段のコンペアスクリプト起動処理の一例である。

【図５】画面への表示内容の一例である。

【図６】タスク向けインストールスクリプトのコマンド記述の一例である。

【図７】コンペア実行手段のコンペアコマンド生成・実行処理の一例である。

【図８】ソフト受信エリアのデータベース構成（プロセスの対象）の一例である。

【図９】プロセス向けインストールスクリプトのコマンド記述の一例である。

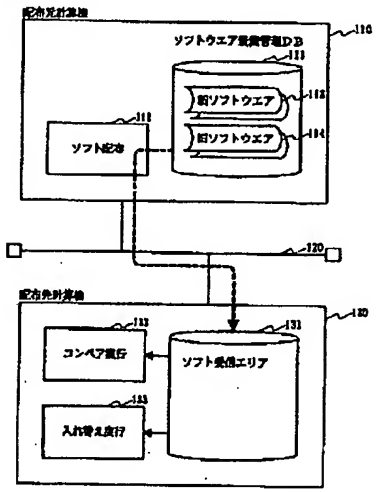
【図１０】コンペア実行手段の仮の実行モジュール生成とコマンド実行処理の一例である。

【符号の説明】

１１０…配布元計算機、１２０…ネットワーク、１３０…配布先計算機、１１１…ソフトウエア資源管理データベース、　１１２…ソフト配布手段、１１３…新バージョンのソフトウエア郡、１１４…旧バージョンのソフトウエア郡、１３１…ソフト受信エリア、１３２…コンペア実行手段、　　１３３…入れ替え実行手段。

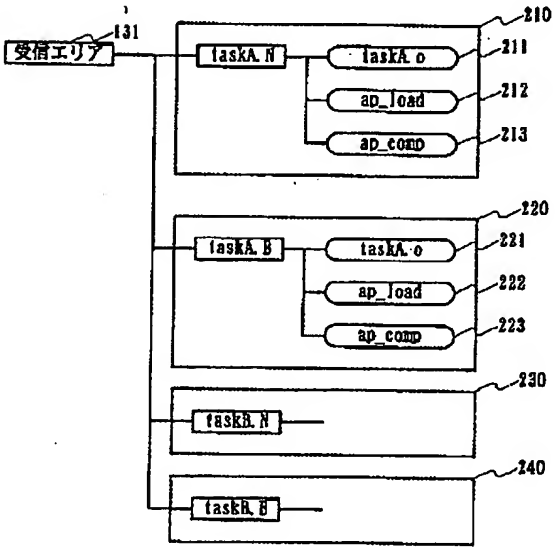
【図１】

図1



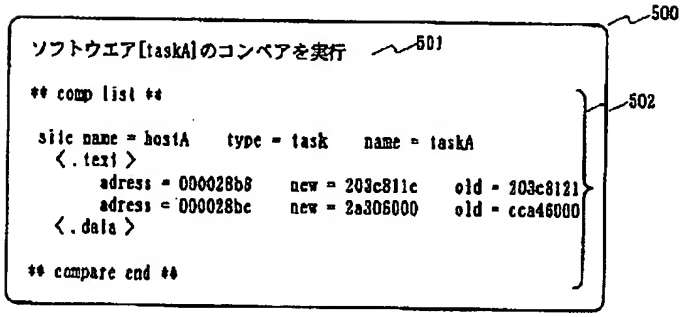
【図２】

図2



【図５】

図5



【図３】

図３

ap_comp

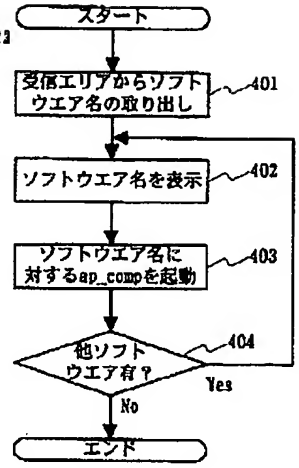
```
#####################################
#  マクロ定義
#####################################
# タスクのコンペア
TASK_COMP()
{
        comp +P -o $1 $2
}
#
#####################################
#   アプリケーション定義
#####################################
#
# タスクのコンペア
#

        TASK_COMP taskA taskA.o
```

222, 302, 303, 301

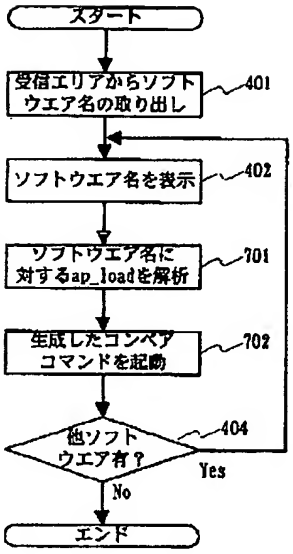
【図４】

図４



【図７】

図７



【図６】

図６

ap_load

```
#####################################
#  マクロ定義
#####################################
# タスクのロード
TASK_LOAD()
{
reload +P $NAME $NO +L $OBJECT -l $LEVEL
}
#
#####################################
#   アプリケーション定義
#####################################
#
# タスクのロード
#
KIND=task
NAME=taskA
NO=100
OBJECT=taskA.o
LEVEL=10

TASK_LOAD
```

222, 603, 604, 601, 602

【図８】

図８